LAW OFFICES OF
FAY KAPLUN & MARCIN, LLP
INTELLECTUAL PROPERTY LAW

150 BROADWAY, SUITE 702
NEW YORK, NEW YORK 10038
PHONE: (212) 619-6000
FAX : (212) 208-6819
WWW.FKMIPLAW.COM

FACSIMILE COVER SHEET

FAX NO	:	(571) 273-8300
TO	:	Mail Stop: Appeal Brief-Patents Commissioner for Patents
FROM	:	Michael J. Marcin, Esq. of Fay Kaplun & Marcin, LLP
DATE	:	May 8, 2008
SUBJECT	:	U.S. Patent Appln. Serial No. 09/920,995 for *System and Method for Implementing a Smart System Call* Our Reference: 40101/08201

NUMBER OF PAGES INCLUDING COVER : 16

MESSAGE:

Please see attached.

Thank you.

IF ANY PAGES WERE NOT RECEIVED OR ARE ILLEGIBLE, PLEASE CALL (212) 619-6000 AS SOON AS POSSIBLE
The information contained in this facsimile message is attorney privileged and confidential information intended only for the use of the individual or entity named above. If the reader of this message is not the intended recipient or the employee or agent responsible to deliver it to the intended recipient, you are hereby notified that any dissemination, distribution or copying of this communication is strictly prohibited. If you have received this communication in error, please notify us by telephone, and return the original message to us at the above address via the U.S. Postal Service. We will reimburse any costs you incur in notifying us and returning the message to us. Thank you.

Attorney Docket No.: 40101/08201 (2000.023)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s)	:	Nunoe
Serial No.	:	09/920,995
Filing Date	:	August 1, 2001
For	:	System and Method for Implementing a Smart System Call
Group Art Unit	:	2194
Confirmation	:	5812
Examiner	:	Charles E. Anya

Mail Stop: Appeal Brief-Patent
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Facsimile
I hereby certify that this correspondence is being deposited via facsimile addressed to:
Mail Stop: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450
571-273-8300

By: Date: May 8, 2008
Michael J. Marcin, (Reg. No. 48,198)

TRANSMITTAL

In response to the Notice of Appeal filed on March 12, 2008, transmitted herewith please find an Appeal Brief for filing in the above-identified application. Please charge the Credit Card of **Fay Kaplun & Marcin, LLP** in the amount of $510.00 (PTO-Form 2038 is enclosed herewith). The Commissioner is hereby authorized to charge the **Deposit Account of Fay Kaplun & Marcin, LLP NO. 50-1492** for any additional required fees. A copy of this paper is enclosed for that purpose.

Respectfully submitted,

Dated: May 8, 2008

By:________________
Michael J. Marcin, Reg. 48,198

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038
Tel: (212) 619-6000
Fax: (212) 619-0276

Attorney Docket No.: 40101/08201 (2000.023)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s)	:	Nunoe
Serial No.	:	09/920,995
Filing Date	:	August 1, 2001
For	:	System and Method for Implementing a Smart System Call
Group Art Unit	:	2194
Confirmation	:	5812
Examiner	:	Charles E. Anya

Mail Stop: Appeal Brief-Patent
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Facsimile

I hereby certify that this correspondence is being deposited via facsimile addressed to:

Mail Stop: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450
571-273-8300

By: Date: May 8, 2008
Michael J. Marcin, (Reg. No. 48,198)

TRANSMITTAL

In response to the Notice of Appeal filed on March 12, 2008, transmitted herewith please find an Appeal Brief for filing in the above-identified application. Please charge the Credit Card of **Fay Kaplun & Marcin, LLP** in the amount of $510.00 (PTO-Form 2038 is enclosed herewith). The Commissioner is hereby authorized to charge the **Deposit Account of Fay Kaplun & Marcin, LLP NO. 50-1492** for any additional required fees. A copy of this paper is enclosed for that purpose.

Respectfully submitted,

Dated: May 8, 2008

By: ________________
Michael J. Marcin, Reg. 48,198

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038
Tel: (212) 619-6000
Fax: (212) 619-0276

Serial No.: 09/920,995
Attorney Docket No.: 40101/08201
Reference No.: 2000.023

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	)	
	)	
Nunoe	)	
	)	
Serial No.: 09/920,995	)	Group Art Unit: 2194
	)	
Filed: August 1, 2001	)	Examiner: Charles E. Anya
	)	
For: SYSTEM AND METHOD FOR IMPLEMENTING A SMART SYSTEM CALL	)))	**Board of Patent Appeals and Interferences**
	)	
Confirmation No.: 5812	)	

Mail Stop: Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF UNDER 37 C.F.R. § 41.37

In support of the Notice of Appeal filed on March 12, 2008, and pursuant to 37 C.F.R. § 41.37, Appellant presents this Appeal Brief in the above-captioned application.

This is an appeal to the Board of Patent Appeals and Interferences from the Examiner's final rejection of claims 1-7 in the Final Office Action dated December 12, 2007. The appealed claims are set forth in the attached Claims Appendix.

1. Real Party in Interest

This application is assigned to Wind River Systems, Inc., the real party in interest.

2. Related Appeals and Interferences

There are no other appeals or interferences that would directly affect, be directly affected, or have a bearing on the instant appeal.

3. Status of the Claims

Claims 1-7 have been rejected in the Final Office Action. The final rejection of claims 1-7 is being appealed.

4. Status of Amendments

All amendments submitted by the Appellant have been entered.

5. Summary of Claimed Subject Matter

The present invention, as recited in independent claim 1, relates to a method that includes determining (102) a current processing mode of an executing software function. (See Specification, p. 6, ll. 22-23; Fig. 1.) When the current processing mode is a privileged processing mode (104), the method executes (106) a direct program flow control instruction to directly access an instruction within a software having the privileged processing mode. (See id., p. 6, l. 23 – p. 7, l. 1; Fig. 1.) When the current processing mode is an unprivileged processing mode (104), the method executes (108) an indirect program flow control instruction to cause execution of the instruction within the software having the privileged processing mode. (See id., p. 7, ll. 1-3; Fig. 1.)

The present invention, as recited in independent claim 6, relates to a method including identifying (202) a program code segment implementing an access to a memory area to be executed within a privileged processing mode. (See id., p. 8, ll. 17-20.) The method also includes replacing the program code segment with a substitute code segment. (See id., p. 12, l. 19 – p. 13, l. 2.) The substitute code segment includes program code to identify (102) a current processing mode of the program code segment. (See id., p. 6, ll. 22-23; Fig. 1.) The program code executes (106) a direct program flow control instruction if the current processing mode is the privileged processing mode (104). (See id., p. 6, l. 23 – p. 7, l. 1; Fig. 1.) The program code executes (108) an indirect program flow control instruction if the current processing mode is an unprivileged processing mode (104). (See id., p. 7, ll. 1-3; Fig. 1.)

The present invention, as recited in independent claim 7, relates to a computer readable medium encoded with a software application, including a software code implementing application functionality and a smart system call into an operating system. The smart system call comprises software code to identify (102) a current processing mode of a program code segment. (See id., p. 6, ll. 22-23; Fig. 1.)The smart system call further comprises software code to execute (106) a direct program flow control instruction if the current processing mode is a privileged processing mode (104). (See id., p. 6, l. 23 – p. 7, l. 1; Fig. 1.) The smart system call further comprises software code to execute (108) an indirect program flow control instruction if the current processing mode is an unprivileged processing mode (104). (See id., p. 7, ll. 1-3; Fig. 1.)

6. Grounds of Rejection to be Reviewed on Appeal

I. Whether claims 1-7 are unpatentable under 35 U.S.C. § 102(e) over U.S. Patent 6,308,255 to Gorishek, IV et al. (hereinafter "Gorishek")

7. Argument

I. The Rejection of Claims 1-7 Under 35 U.S.C. § 102(e) Should Be Reversed.

A. The Examiner's Rejection

In the Final Office Action, the Examiner rejected claims 1-7 as anticipated by Gorishek. (See 12/12/07 Office Action, pp. 2-4.) This rejection was reaffirmed in the Advisory Action. (See 2/21/08 Advisory Action, pp. 2-3.)

B. Gorishek Does Not Disclose Executing a Direct Program Flow Control Instruction To Directly Access an Instruction Within a Software Having The Privileged Processing Mode and Executing an Indirect Program Flow Control Instruction To Cause Execution of the Instruction Within the Software Having the Privileged Processing Mode As Recited In Claim 1

Claim 1 recites "[a] method, comprising: determining a current processing mode of an executing software function; when the current processing mode is a privileged processing mode, executing a direct program flow control instruction to directly access an instruction within a software having the privileged processing mode; and when the current processing mode is an unprivileged processing mode, executing an indirect program flow control instruction to cause execution of the instruction within the software having the privileged processing mode."

Gorishek generally relates to a computer system including a host processor and an emulation coprocessor. (See Gorishek, Abstract). According to Gorishek, the "host processor" comprises hardware configured to execute instructions defined by a host instruction set architecture, while the "emulation coprocessor" comprises hardware configured to execute instructions defined by a different (or foreign) instruction set architecture. (See id., col. 6, ll. 6-28.) Accordingly, the host processor executes operating system code as well as application programs that are coded in the host instruction set architecture, while the emulation coprocessor executes the foreign application program. (See id.) Therefore, when a user submits a command to initiate an application program, the system examines the file format of the application program in order to determine what type of code is included in the application program. (See id., col. 13, ll. 44-49.) If *the application program* is determined to be coded according to the host instruction set architecture, the system establishes the process as a normal host process and the application is executed by the host processor. (See id., col. 13, ll. 44-49.) Alternatively, if *the application program* is determined to be coded according to a foreign instruction set architecture executable by the emulation coprocessor, then the system invokes the emulation coprocessor interface code in order to initiate the foreign application program. (See id., col. 13, l. 55 – col. 14, l. 2.) Thus, the initiation of an application program according to the system and method disclosed by Gorishek is dependent on the application program, specifically the code within the application program. In other words, the system according to Gorishek determines which of the two processors (the host or the emulation coprocessor) will execute the code of an application program depending on the format of the code. However, the system according to Gorishek fails to teach or suggest that *a particular instruction within a software having the privileged processing mode* may be executed via a direct program flow control instruction (when the processing mode is privileged) *and* may also be executed via an indirect program flow control instruction (when the processing mode is unprivileged), as recited in claim 1. In fact, since the application program described in Gorishek is executed by the host processor *or* the emulation coprocessor, Gorishek teaches away from recitations of claim 1.

In the Response to Arguments, the Examiner asserts that the current processing mode by Gorishek is determined by examining the file format of the application program, and the system will execute a normal host process *if* the file format is in accord with the host instruction set. (See 12/12/07 Office Action, p. 5, ll. 6-20.) The Examiner continues to state that the system of Gorishek would otherwise execute a foreign application program via the thunk/emulation coprocessor. (See id.) While the Applicant does not concede that Gorishek discloses "determining a current processing mode of an executing software function," it should be noted the that the Examiner fails to demonstrate that Gorishek discloses an instruction within a software having privileged processing mode is executable by *both* a direct program flow control instruction *and* an indirect program flow control instruction, as recited in claim 1. Seeing as the system and method disclosed by Gorishek is dependent on the code of the application program requested by the user, an application program that includes code formatted for the host processor can *only* be executed by the host processor. Likewise, an application program that includes code according to its foreign instruction set architecture can *only* be executed by the emulation coprocessor. Accordingly, a specific application program is executable by *either* the host processor *or* the emulation coprocessor, but not executable in two different modes. Therefore, as discussed above, the host processor of the Gorishek system is configured to execute one set of instructions while the emulation coprocessor is configured to execute a different set of instructions. (See Gorishek, col. 6, ll. 16-28.)

In contrast to Gorishek, the method recited in claim 1 of the present invention is not dependent on the file format of an instruction within a software. However, the manner in which the instructions recited in claim 1 are executed is dependent on the current processing mode. It is clear within the claim language that an instruction within the software may be executed directly when the processing mode is privileged, and the same instruction within the software may be executed indirectly when the processing mode is unprivileged. Accordingly, the instructions within the software, described in claim 1, may be executed in both processing modes, regardless of the file format of the instructions. The same cannot be said for the system described by Gorishek. As discussed above, and as reiterated by the Examiner, Gorishek examines the file format in order to have the application program executed by the host processor or executed by the emulation coprocessor. (See 12/12/07 Office Action, p. 5, ll. 16-20.)

Thus, it is respectfully submitted that Gorishek does not disclose nor suggest, a "method, comprising: determining a current processing mode of an executing software function; when the current processing mode is a privileged processing mode, executing *a direct program flow control instruction to directly access an instruction within a software having the privileged processing mode*; and when the current processing mode is an unprivileged processing mode, executing *an indirect program flow control instruction to cause execution of the instruction within the software having the privileged processing mode*," as recited in claim 1. Accordingly, Applicants respectfully submit that for at least the reasons stated above, claim 1 of the present application is not anticipated by Gorishek, and request that the rejection of this claim be overturned. As claims 2-5 depend from, and therefore include all the limitations of claim 1, it is hereby submitted that these claims are also allowable.

Claim 6 recites, "[a] method, comprising: identifying a program code segment implementing an access to a memory area to be executed within a privileged processing mode; replacing the program code segment with a substitute code segment; wherein the substitute code segment includes program code to identify a current processing mode of the program code segment, *execute a direct program flow control instruction if the current processing mode is the privileged processing mode, and execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode.*" The Appellant respectfully submits that the rejection of claim 6 should be overturned for at least the reasons discussed above with reference to claim 1.

Claim 7 recites, "[a] computer readable medium encoded with a software application, comprising: a software code implementing application functionality; and a smart system call into an operating system; wherein the smart system call comprises the software code to identify a current processing mode of a program code segment, *execute a direct program flow control instruction if the current processing mode is a privileged processing mode, and execute* *an indirect program flow control instruction if the current processing mode is an unprivileged processing mode*." The Appellant respectfully submits that the rejection of claim 7 should be overturned for at least the reasons discussed above with reference to claim 1.

8. Conclusion

For the reasons set forth above, the Appellant respectfully requests that the Board reverse the rejection of the claims by the Examiner under 35 U.S.C. § 102(e), and indicate that claims 1-7 are allowable.

Respectfully submitted,

Date: May 8, 2008

By: ______________________
Michael J. Marcin (Reg. No. 48,198)

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038
Tel.: (212) 619-6000
Fax: (212) 619-0276

CLAIMS APPENDIX

1. (Previously Presented) A method, comprising:

determining a current processing mode of an executing software function;

when the current processing mode is a privileged processing mode, executing a direct program flow control instruction to directly access an instruction within a software having the privileged processing mode; and

when the current processing mode is an unprivileged processing mode, executing an indirect program flow control instruction to cause execution of the instruction within the software having the privileged processing mode.

2. (Original) The method of claim 1, wherein the direct program flow control instruction is a jump instruction.

3. (Original) The method of claim 1, wherein the indirect program flow control instruction is an interrupt instruction.

4. (Original) The method of claim 1, wherein the software having the privileged processing mode is operating system software.

5. (Original) The method of claim 4, wherein the software having the privileged processing mode is kernel software.

6. (Original) A method, comprising:

identifying a program code segment implementing an access to a memory area to be executed within a privileged processing mode;

replacing the program code segment with a substitute code segment;

wherein the substitute code segment includes program code to

identify a current processing mode of the program code segment,

execute a direct program flow control instruction if the current processing mode is the privileged processing mode, and execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode.

7. (Previously Presented) A computer readable medium encoded with a software application, comprising:

a software code implementing application functionality; and

a smart system call into an operating system;

wherein the smart system call comprises the software code to

identify a current processing mode of a program code segment,

execute a direct program flow control instruction if the current processing mode is a privileged processing mode, and

execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode.

EVIDENCE APPENDIX

No evidence has been submitted herewith or is relied upon in the present appeal.

RELATED PROCEEDINGS APPENDIX

No decisions have been rendered regarding the present appeal or any proceedings related thereto.